**House Price Prediction**

# In this project , I used the XGBoost Regressor along with [RFECV](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html#sklearn.feature_selection.RFECV)  (Recursive Feature Elimination) to rank the importance of the features and eliminate the redundant ones. I then trained a Kernel Ridge regressor (which performed best among the several models I tried) to make predictions, which leads to a [leaderboard score](https://www.kaggle.com/c/house-prices-advanced-regression-techniques/leaderboard) of 0.11709.

The steps

## -Data Pre-processing

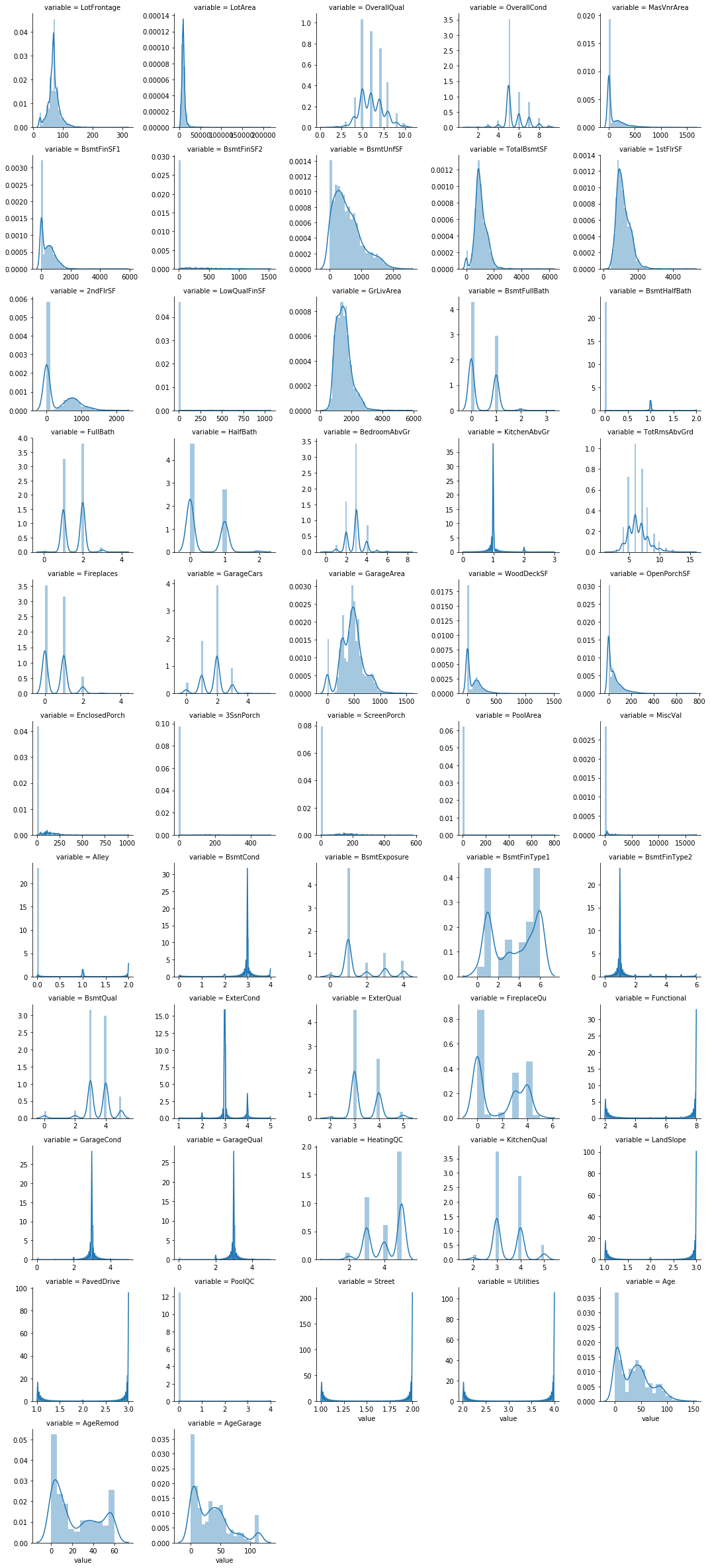
-divide the data into numerical ("quan") and categorical ("qual") features

-Find out how many missing values there are for the quantitative and categorical features

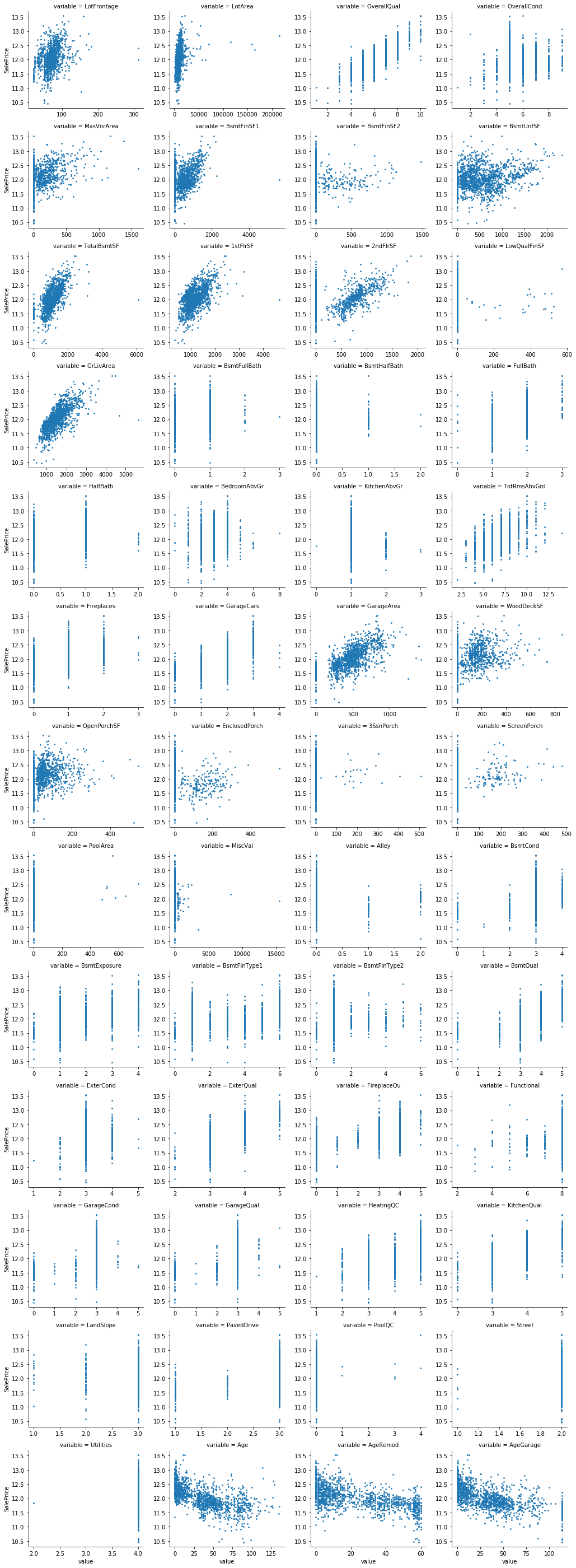
-Filling missing values

-These categorical features are "rank", so they can be transformed to numerical features

-After the cleaning data we do visualize the distribution of each numerical feature



After that we scatter plots



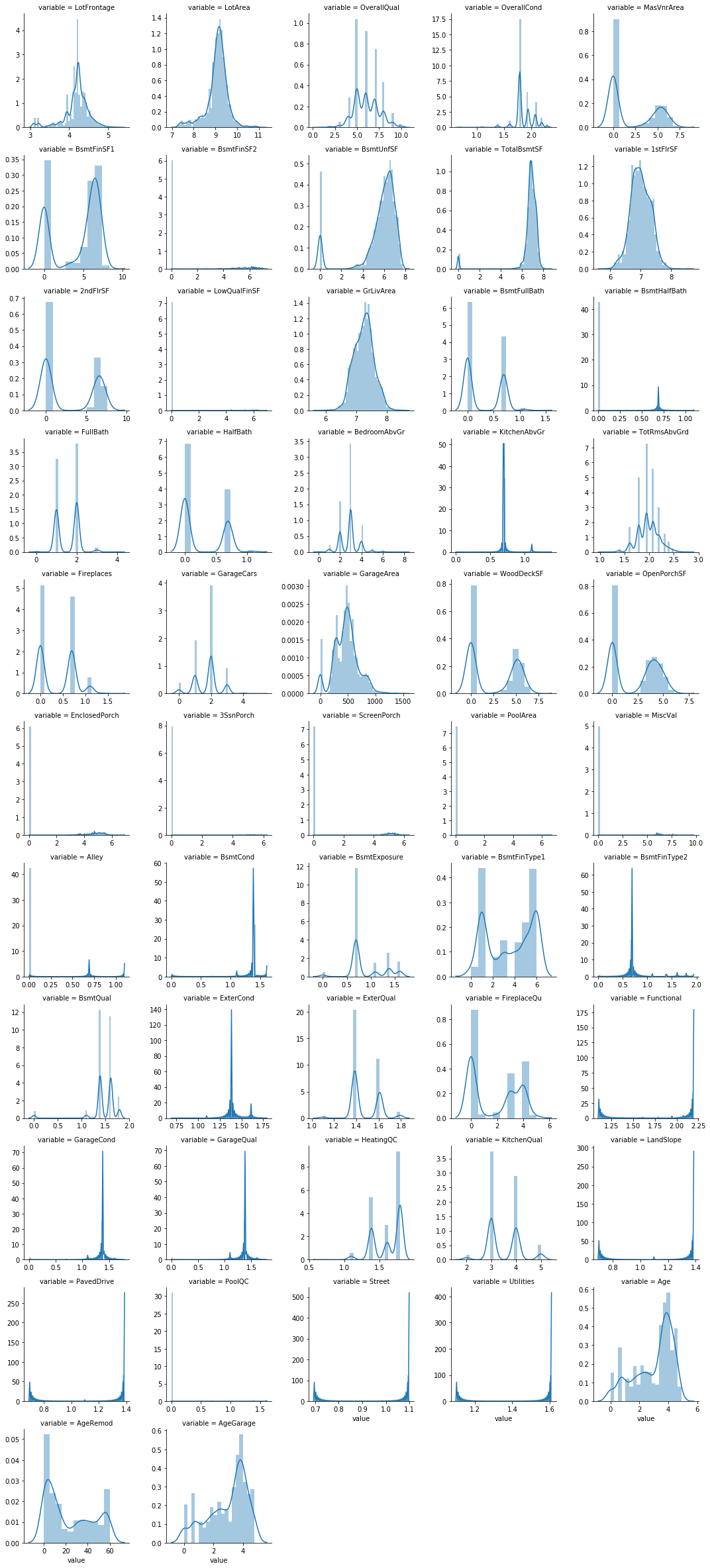
We see correlations in the scatter plots for some features, such as LotFrontage, LotArea, TotalBsmtSF...etc. There are some obvious outliers in some of these plots. By removing some outliers in the training set, the model will generalize better on unseen data.

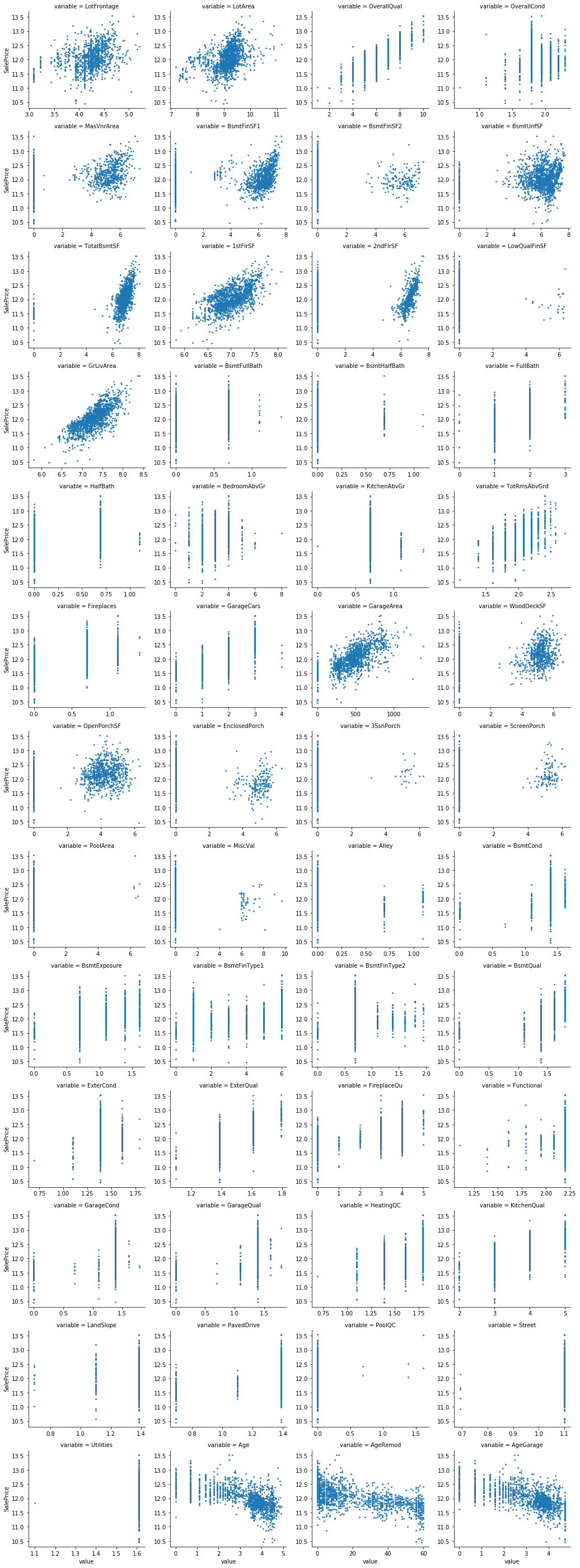
After removing the outliers it is necessary to transform the numerical features that are skewed. This is because lots of regression models building assume that the features are distributed normally and have a symmetrical shape.

I've tried various transformations (log, boxcox, sqrt...etc) and found that log-transform works better.

And we transform those with skewness > 0.5

We get the data look like this

scatter plots look like this



For the categorical features, I will transform them to dummy variables, but I'll drop one column from each of them to avoid [dummy variable trap](http://www.algosome.com/articles/dummy-variable-trap-regression.html).

After that I can split the data into training set and test set. And then I'll perform Standardization on the numerical features (those that are not dummy variables). Here I use RobustScaler instead, which is more robust to outliers in the data.

Now there are 220 features due to a large amount of the dummy variables. Overfitting can easily occur when there are redundant features, which also leads to longer computation time. Therefore, I use XGBoost regressor to generate the rank of "feature importance"

In the end we can use [RFECV](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html#sklearn.feature_selection.RFECV) to eliminate the redundant features.

Those are the most importance feature we get after we use RFECV

array(['1stFlrSF', 'BsmtExposure', 'BsmtFinSF1', 'BsmtFinType1',

'BsmtQual', 'ExterQual', 'FireplaceQu', 'Functional', 'GarageArea',

'GarageCars', 'GarageCond', 'GarageQual', 'GrLivArea', 'HalfBath',

'KitchenAbvGr', 'KitchenQual', 'LotArea', 'OverallCond',

'OverallQual', 'TotalBsmtSF', 'WoodDeckSF', 'Age', 'AgeRemod',

'MSZoning\_C (all)', 'MSZoning\_RL', 'MSZoning\_RM',

'Neighborhood\_Crawfor', 'Neighborhood\_OldTown',

'Neighborhood\_Sawyer', 'Condition1\_Artery', 'Exterior1st\_BrkComm',

'Heating\_Grav', 'CentralAir\_Y', 'GarageType\_Attchd',

'GarageType\_Detchd', 'SaleType\_New', 'SaleCondition\_Abnorml',

'MSSubClass\_class2'],

## Modeling and Prediction

Kernel Ridge regressor is the model that performed best among the several models (XGBoost, Neural Network, Random Forest...etc) we have tried. Here I tune its hyperparameters using RandomizedSearchCV

Parameters of the best\_estimator:

{'alpha': 0.44725674716804203, 'coef0': 3.9974729787275987, 'degree': 2, 'kernel': 'polynomial'}

Mean cross-validated RMSE of the best\_estimator: 0.11351751593503093

RMSE of the whole training set: 0.09933518141866216